EASE: BEHAVIOR-ADAPTIVE SKILL CURATION FOR SELF-EVOLVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent skills provide a lightweight mechanism for self-evolving agents to accumulate reusable procedural knowledge without updating model parameters. However, existing learned skill curators typically optimize curation without explicitly modeling the behavior of the downstream executor. We show that this limitation can lead to systematic cross-executor degradation: curators optimized with different executors perform best when paired with their own training executor, suggesting that effective skill curation is executor-dependent. To address this challenge, we formulate behavior-adaptive skill curation and introduce **EASE**, a framework that learns a single curator capable of adapting its curation decisions to different executor behaviors. EASE maintains an online behavioral profile summarizing recent execution patterns and conditions the curator on this profile, together with the current trajectory and retrieved skills, to dynamically add, modify, or remove skills from an evolving repository. We train the shared curator jointly across multiple frozen executors with reinforcement learning, using retrieval-aware and behavior-aware temporal attribution to focus optimization on curation actions with observable downstream influence. Across multiple public agentic benchmarks, including ALFWorld, ScienceWorld, and WebShop, and heterogeneous executor families ranging from Qwen3-8B/32B and GPT-OSS-120B to previously unseen Kimi K2.6, DeepSeek V4 Flash, and Gemini 3.5 Flash, EASE outperforms strong skill- and memory-based baselines without per-executor finetuning. Beyond task performance, behavior-adaptive curation also enables more efficient skill evolution: across the three benchmarks, EASE maintains $34.5{\sim}41.0\%$ fewer skills, improves skill retrieval by $36.3{\sim}38.7\%$ and measured edit utility by $51.8{\sim}60.0\%$, and reduces deployment-time inference tokens by $9.1{\sim}14.5\%$. These results establish behavior-adaptive skill curation as an effective principle for building self-evolving agents. Anonymized code is available at `https://anonymous.4open.science/r/EASE-skills`.

1 INTRODUCTION

Large language model (LLM) agents increasingly operate in interactive environments that require long-horizon reasoning, tool use, and continual adaptation from experience (Yang et al., 2024; Hu et al., 2025). While model-based finetuning can improve an agent over time, repeatedly updating model parameters during deployment is often impractical. The executor may be accessible only through an API, optimization can be computationally expensive, and continual weight updates may introduce instability or overfitting. A lightweight alternative is to accumulate experience as agent skills, namely compact and reusable descriptions of procedural knowledge that can be retrieved and reused in future tasks without modifying the executor itself (Anthropic, 2025; Ma et al., 2026). Recent work has therefore explored automatically constructing and curating evolving skill repositories from agent trajectories and interaction feedback (Xia et al., 2026; Ni et al., 2026; Wang et al., 2026a;b; Ouyang et al., 2026a).

A key challenge, however, is that effective skill curation may not be independent of the executor that consumes those skills. Different executors can exhibit different action preferences, failure modes, recovery behaviors, reasoning patterns, and abilities to exploit retrieved instructions. As a result, the same skill or curation decision may not have the same downstream utility across executors. Existing learned skill curators (Wang et al., 2026b; Ouyang et al., 2026a) typically optimize curation without

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

explicitly modeling such executor-specific behavior, which can make their decisions brittle when the deployment executor changes.

We observe this phenomenon empirically in Figure 1. When separately trained skill curators are paired with different executors, each curator performs best with the executor on which it was optimized, while mismatched curator and executor pairs consistently degrade. This pattern suggests that effective skill curation is executor-dependent. The curator should not only reason about what experience has been collected, but also about how the downstream executor is likely to use that experience.

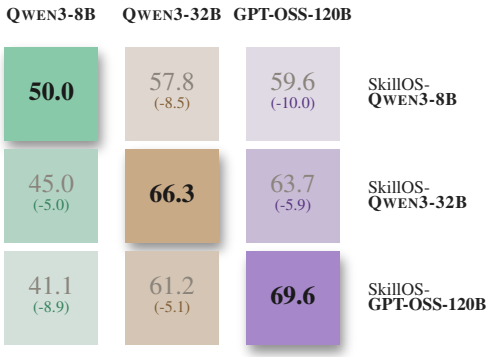


Figure 1: **Skill curation is executor-dependent.** We independently train a skill curator with each executor and evaluate all curator–executor pairings. Matched pairs consistently outperform mismatched ones, suggesting that effective skill curation depends on downstream executor behavior and motivating behavior-adaptive skill curation.

This motivates a distinct problem that we call *behavior-adaptive skill curation*: learning a shared curator that dynamically adjusts its curation decisions according to the observed behavior of the executor that will consume the resulting skills. A straightforward solution would be to train a dedicated curator for each executor, but this is costly and cannot support newly released or otherwise unseen executors. Pooling data from multiple executors is also insufficient by itself, because the same curation decision may receive conflicting supervision under different executor behaviors. Conditioning on model identity or static metadata may help for known executors, but provides limited information about how an unseen executor actually behaves during interaction. We therefore argue that adaptation should be driven by behavioral evidence inferred online from execution itself.

To address this challenge, we introduce EASE, a framework for behavior-adaptive skill curation in self-evolving agents. EASE continuously summarizes an executor's recent interaction patterns into an online behavioral profile. A shared curator conditions on this profile, together with the current trajectory and retrieved skills, to dynamically add, modify, or remove skills from the evolving repository. We train the curator jointly across multiple frozen executors with reinforcement learning, encouraging it to learn how curation decisions should vary with executor behavior rather than overfitting to any single executor. EASE further uses retrieval- and behavior-aware temporal credit assignment to focus optimization on curation actions with observable downstream influence. At deployment, both curator and executor parameters remain fixed, while adaptation proceeds through updates to the behavioral profile and skill repository. We evaluate EASE across multiple interactive agent benchmarks and heterogeneous executor families, including both executors seen during curator training and previously unseen executors. A single EASE curator matches or improves executor-specific curators on seen executors and generalizes to unseen ones without per-executor finetuning. Beyond task performance, behavior-adaptive curation also produces more compact and useful skill repositories, with higher skill retrieval and edit utility as well as lower deployment-time inference overhead. These results suggest that explicitly adapting skill curation to executor behavior provides a more robust foundation for self-evolving agents whose underlying executors may change over time. Our main **contributions** are summarized as follows:

- **Behavior-adaptive skill curation.** We identify a fundamental challenge in self-evolving skill curation: effective curation is not executor-independent, but changes with the behavior of the downstream executor. We formulate behavior-adaptive skill curation as a new paradigm in which skill evolution explicitly conditions its curation decisions on observed executor behavior, rather than assuming that a fixed curation policy transfers uniformly across executors.

- **EASE.** We introduce a shared behavior-adaptive skill curator that continuously summarizes recent execution patterns into an online behavioral profile and uses this profile to guide skill addition, modification, and deletion. EASE is trained jointly across multiple frozen executors with reinforcement learning, together with retrieval- and behavior-aware temporal credit assignment that focuses optimization on curation actions with observable downstream influence.

- **Broad empirical advantage.** We show that EASE generalizes across heterogeneous executors and multiple interactive environments, including executors unseen during curator training, without per-executor finetuning. Beyond task performance, EASE produces more compact and useful skill repositories, with higher skill retrieval and edit utility and lower deployment-time inference overhead, demonstrating that behavior adaptation improves both the effectiveness and efficiency of self-evolving skill curation.

2 Related Work

Automated agent skills. Prior work enables agents to reuse past experience through reflections, reusable insights, executable skills, workflows, and procedural memories (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023; 2025; Zhang et al., 2026b; Ouyang et al., 2026b; Fang et al., 2026). Building on this direction, recent methods increasingly automate the construction and refinement of reusable skills from execution trajectories and feedback (Ni et al., 2026; Wang et al., 2026a; Alzubi et al., 2026; Ma et al., 2026; Zhang et al., 2026a). Several approaches further study whether generated skills are actually useful for downstream execution, for example by evaluating their marginal contribution or repairing low-utility skills (Ma et al., 2026; Wang et al., 2026c). A complementary line of work moves from optimizing individual skill artifacts to learning policies for skill generation and curation, often through reinforcement learning over sequential interactions (Wang et al., 2026b; Ouyang et al., 2026a; Zhang et al., 2026c; Xia et al., 2026). EASE builds on this latter direction, but focuses on a different question: how should a learned curator adapt its decisions when the executor that consumes the resulting skills exhibits different execution behavior? To support such adaptation, EASE conditions curation on an online behavioral profile and uses temporal credit assignment to focus learning on curation actions with observable downstream influence.

Skill transferability and executor adaptation. Recent work also investigates whether learned skills remain useful when transferred across agents or model backbones. Trace2Skill, SkillGen, and CoEvoSkills evaluate generated skills on models different from those used to construct them (Ni et al., 2026; Ma et al., 2026; Zhang et al., 2026a), showing that reusable skill artifacts can transfer across executors, while also revealing that their effectiveness may remain model-dependent. At the policy level, SkillOS studies whether a learned skill curator can transfer across frozen executor backbones (Ouyang et al., 2026a). These studies establish cross-executor transfer as an important capability, but transferability alone does not explicitly model how curation should change with the behavior of the deployment executor. EASE instead formulates behavior-adaptive skill curation: a single shared curator observes an executor's online behavioral profile and adjusts its add, modify, and delete decisions accordingly. This formulation allows adaptation to both seen and previously unseen executors without executor-specific fine-tuning, treating cross-executor generalization as a consequence of behavior-conditioned curation rather than the primary objective.

3 Method

We present EASE, a framework for behavior-adaptive skill curation in self-evolving agents (Figure 2). Instead of training a separate curator for each executor, EASE adapts skill curation to the observed behavior of the executor that consumes the resulting skills.

3.1 Behavior-Adaptive Skill Curation and Framework Overview

We consider a self-evolving agent composed of a frozen LLM-based executor E, an evolving skill repository $\mathcal{S}_t$, and a shared skill curator π_θ. Following Ouyang et al. (2026a), tasks $x_1, x_2, \ldots$ arrive sequentially, and each task is completed before the next one is observed. Before task x_t, the executor retrieves at most k relevant skills $\mathcal{S}_t^{\mathrm{ret}} = \mathcal{R}_k(x_t; \mathcal{S}_t)$. The executor then interacts with the environment, producing an agentic trajectory ξ_t and, when available, feedback c_t from the environment or its own self-assessment. After the task, the curator emits curation operations Δ_t that update the repository to $\mathcal{S}_{t+1} = \Delta_t(\mathcal{S}_t)$, which is available only to subsequent tasks.

Behavior-adaptive curation. Existing learned curators decide Δ_t from the current experience and retrieved skills alone, $\pi_\theta(\cdot \mid \xi_t, c_t, \mathcal{S}_t^{\mathrm{ret}})$, without representing how the downstream executor con-

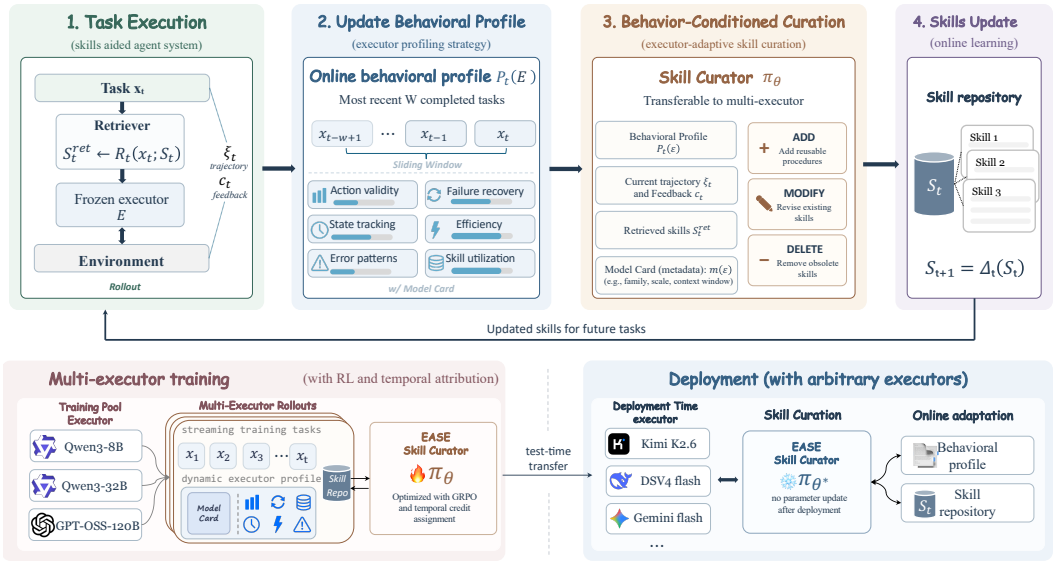


Figure 2: **Overview of EASE framework.** *Top:* the online loop for one executor. (1) A frozen executor E solves task x_t with skills retrieved from the repository $\mathcal{S}_t$. (2) The resulting trajectory and feedback update the online behavioral profile $P_t(E)$, which summarizes execution statistics over recent W tasks. (3) Conditioned on the executor descriptor $e_t = [P_t(E); m(E)]$, the current trajectory, and the retrieved skills, a skill curator π_θ performs different operations. (4) The updated repository $\mathcal{S}_{t+1}$ serves subsequent tasks. *Bottom left:* π_θ is trained with GRPO across a pool of frozen executors with temporal-attribution gates. *Bottom right:* at deployment, the frozen curator adapts to unseen executors without per-executor finetuning.

sumes the resulting skills. Since curator effectiveness varies systematically across executors (Figure 1), we formulate behavior-adaptive skill curation, in which the curator additionally conditions on a behavioral representation of the executor:

$$\Delta_t \sim \pi_\theta\big(\cdot \mid \xi_t, c_t, \mathcal{S}_t^{\text{ret}}, b_t(E)\big)\,, \tag{1}$$

where $b_t(E)$ summarizes the executor's behavior observed up to task t. Rather than identifying the executor and selecting an executor-specific curator, EASE learns a single conditional policy whose decisions vary with observable behavior, which also applies to previously unseen executors. We instantiate $b_t(E)$ with an online behavioral profile $P_t(E)$, described next.

3.2 Online Executor Behavioral Profiling and Skill Curation

Accumulating behavioral evidence. A single trajectory describes one execution, whereas behavior-adaptive curation requires evidence about recurring execution patterns. For each executor and task stream, let $P_t(E)$ denote the behavioral profile after completing task x_t. We aggregate evidence over a sliding window containing the most recent W completed tasks:

$$P_t(E) = \text{Agg}\Big(\big\{\phi(\xi_i, c_i, \mathcal{S}_i^{\text{ret}})\big\}_{i=\max(1,t-W+1)}^{t}\Big)\,, \qquad P_0(E) = \varnothing. \tag{2}$$

Here, ϕ extracts observable event counts and measurements, while Agg combines them into counts, rates, and averages together with their sample sizes. Before W tasks have been completed, the profile uses all available episodes. The finite window allows the profile to track the executor's recent behavior as the skill repository evolves, rather than treating behavior as a fixed intrinsic property of the underlying model.

The feedback c_i is the executor's self-assessment or feedback exposed by the environment and does not require access to held-out answers. The profile covers action validity, failure recovery, state tracking, execution efficiency, observable error patterns, and skill utilization. Features are computed from logged interactions without additional model calls. Missing signals are treated as unavailable

rather than recorded as zero. In particular, retrieval is directly observable, whereas implicit reliance on a skill may not always be identifiable. We therefore distinguish retrieval from explicit usage evidence. Appendix B provides the detailed feature families and aggregation rules.

Behavior-conditioned curation. After the current trajectory is incorporated into $P_t(E)$, the curator samples a sequence of curation operations:

$$\Delta_t \sim \pi_\theta\big(\cdot \mid \xi_t, c_t, \mathcal{S}_t^{\mathrm{ret}}, e_t\big), \qquad e_t = [P_t(E); m(E)], \qquad \mathcal{S}_{t+1} = \Delta_t(\mathcal{S}_t), \tag{3}$$

where $m(E)$ denotes optional static executor metadata.

The curator supports three repository operations: `add`, `modify`, and `delete`, corresponding to inserting a new reusable skill, revising an existing skill, and removing a redundant or counterproductive skill, respectively. The online behavioral profile provides the primary adaptation signal. For example, repeated action rejections may motivate skills with explicit precondition checks, whereas an executor that already handles such constraints reliably may benefit from shorter or higher-level procedural guidance.

When available, static metadata $m(E)$, such as model family, scale, or context window, provides supplementary context. However, it does not replace behavioral evidence and unavailable fields are explicitly marked as unknown. This distinction is important because online behavioral profiles can be constructed for both seen and previously unseen executors from their actual interactions.

At cold start, $P_0(E)$ and the repository are empty. The first curation decision therefore uses the profile formed after the first completed task. Profiles and repositories are maintained independently across executors and rollouts and are never shared between independent streams.

3.3 Learning a Behavior-Adaptive Curator

A behavior-conditioned curator is useful only if it learns how different behavioral patterns should induce different curation decisions. We therefore train only the curator, using experience collected from a pool of frozen executors. For each training group, we sample an executor E and an ordered stream $G = (x_1, \ldots, x_T)$ of $T \geq 2$ related tasks, and collect N independent rollouts with the same executor and task sequence, each maintaining its own skill repository, behavioral profile, and execution history. Related tasks allow skills acquired earlier in the stream to influence later tasks, while the streaming protocol prevents the curator from observing future tasks when making a curation decision.

Holding the executor and task sequence fixed within a group makes group-relative reward comparisons conditional on the same execution setting, while varying the executor across groups exposes the curator to different behavioral patterns and skill needs. Multi-executor training is therefore not merely a way to increase data diversity: its purpose is to teach a shared curator how the mapping from execution experience to appropriate curation changes with executor behavior (Appendix C discusses the remaining sources of variation).

Reward and group-relative advantage. We optimize the curator with GRPO (Shao et al., 2024). Following the reward components of Ouyang et al. (2026a), rollout n receives

$$R^{(n)} = r_{\mathrm{task}}^{(n)} + \lambda_f r_{\mathrm{func}}^{(n)} + \lambda_c r_{\mathrm{comp}}^{(n)} + \lambda_u r_{\mathrm{qual}}^{(n)}, \qquad A^{(n)} = \frac{R^{(n)} - \mu_G}{\sigma_G + \epsilon_A}, \tag{4}$$

where μ_G and σ_G denote the mean and standard deviation across the N rollouts in the same group, and $\epsilon_A > 0$ stabilizes normalization. The task reward averages outcomes on $x_2, \ldots, x_T$ (x_1 is excluded because its repository is initially empty), and the auxiliary terms, averaged over curation steps, reward successfully executed curation calls, repository compactness, and content quality assessed by a training-time judge. We exclude descriptor tokens from the compression denominator so that a longer behavioral profile or model card does not itself increase the compression reward (Appendix F).

3.4 Retrieval- and Behavior-Aware Temporal Credit Assignment

The rollout-level advantage evaluates the evolving repository as a whole, but does not identify which individual curation actions plausibly influenced later execution; applying it to every edit therefore

introduces noisy supervision, especially when an edited skill is never retrieved again. To address this temporal credit assignment problem, EASE uses two approximate temporal-attribution gates to select curation actions with observable downstream influence (exact definitions and edge cases in Appendix D).

Gate 1: retrieval incidence. For an edit at step t, let Z_t count subsequent tasks for which a skill touched by Δ_t either enters retrieval or would have entered retrieval had the edit not been applied. To account for `delete` and `modify` operations, the test also considers a patched repository that restores the pre-edit skill state. Gate 1 admits an edit only if $Z_t > 0$, excluding curation actions with no observed opportunity to influence later execution.

Gate 2: behavioral deviation. Retrieval alone does not establish that a skill affects executor behavior. For each retrieval-incident edit, we therefore compare subsequent skill-assisted action sequences against cached skill-free reference trajectories generated under the same executor and environment configuration, computing a normalized Levenshtein deviation while subtracting background variation among the skill-free runs. Let $\mathcal{D}_j$ denote this deviation on task j. Gate 2 admits an edit only if at least one later retrieval-incident task satisfies $\mathcal{D}_j > \tau$ for a threshold $\tau \geq 0$. With both gates enabled, the admitted set for rollout n is

$$\mathcal{M}^{(n)} = \left\{ t : Z_t^{(n)} > 0 \text{ and } \max_{j>t:\ \text{touched skill retrieved at } j} \mathcal{D}_j^{(n)} > \tau \right\}; \tag{5}$$

variants that disable one or both gates are defined in Appendix D. Because $\mathcal{D}_j$ measures whether skill-conditioned execution departs from the executor's skill-free behavior, rather than the marginal contribution of a particular edit, we interpret the two gates as approximate temporal attribution, rather than exact causal credit assignment.

Masked policy optimization. Let $q_{t,\ell}^{(n)}(\theta)$ denote the token-level importance ratio for token ℓ at curation step t, and let $A^{(n)}$ denote the rollout-level advantage. We maximize the clipped PPO surrogate only over admitted curation steps:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{I}|} \sum_{n \in \mathcal{I}} \frac{1}{|\mathcal{M}^{(n)}|} \sum_{t \in \mathcal{M}^{(n)}} \frac{1}{L_t^{(n)}} \sum_{\ell=1}^{L_t^{(n)}} \ell_{\text{clip}} \left(q_{t,\ell}^{(n)}(\theta), A^{(n)} \right), \tag{6}$$

where $\mathcal{I} = \{n : |\mathcal{M}^{(n)}| > 0\}$ and ℓ_{clip} is the standard PPO clipped surrogate. Advantages are computed from all N rollouts before masking, so the rollout-level reward determines whether a trajectory is beneficial overall, while temporal attribution determines which curation actions receive policy-gradient supervision; Appendix D details the handling of empty admitted sets and fixed masks.

Deployment-time adaptation. The quality judge, skill-free reference trajectories, and temporal-attribution gates are used only during training. At deployment, curator and executor parameters remain fixed, and EASE adapts solely through the behavioral profile $P_t(E)$ and the skill repository $\mathcal{S}_t$, which allows the same curator to serve executors that were not available during training.

4 Experiments

We organize our experiments around four research questions (**RQ**s). **RQ1**: Can a single curator serve many executors, including ones it has never seen? **RQ2**: Does adapting online at deployment actually matter? **RQ3**: Does behavior-aware curation make skills more useful and cheaper to use? **RQ4**: Which components make our behavior-aware curation work?

4.1 Experimental Settings

Benchmarks. We evaluate on three text-based interactive benchmarks: ALFWorld (Shridhar et al., 2021), a household environment with six task categories; ScienceWorld (Wang et al., 2022), a simulated laboratory whose tasks require multi-step scientific procedures such as measuring, heating, and mixing; and WebShop (Yao et al., 2022), a simulated e-commerce website where the agent searches for and purchases products that match natural-language instructions.

Metric. On ALFWorld, we report success rate (SR) macro-averaged over task categories, so that each category contributes equally regardless of its size (Eq. 7, Appendix A). On ScienceWorld and WebShop, we report the standard benchmark score.

Executors. The curator is trained with three frozen executors, $\mathcal{E}_{\text{train}} = \{$Qwen3-8B, Qwen3-32B, GPT-OSS-120B$\}$ (Qwen Team, 2025; OpenAI, 2025). To test transfer, we evaluate with three API executors from other model families, Kimi K2.6, DeepSeek V4 Flash, and Gemini 3.5 Flash, which are excluded from both training and checkpoint selection.

Baselines. *Training-free* baselines keep all model weights fixed: No Memory runs the executor without any memory or skills; AWM (Wang et al., 2025), ReasoningBank (Ouyang et al., 2026b), MemP (Fang et al., 2026), and Memento-Skills (Zhou et al., 2026) augment the executor with workflows, reasoning memories, procedural memories, or skills accumulated from experience. *Finetuned curators*: SkillOS (Ouyang et al., 2026a) trains a skill curator with RL for a single executor; we denote the three seen executors by α (Qwen3-8B), β (Qwen3-32B), and γ (GPT-OSS-120B), so that SkillOS-α is the curator trained with Qwen3-8B, and so on. SkillOS-base denotes the same curator without RL training. For unseen executors, we run all three transferred curators and report, for each executor and benchmark, the highest score among them (SkillOS-*best*).

Implementation. The curator is Qwen3-8B finetuned with LoRA (Hu et al., 2022) and GRPO (Shao et al., 2024); at evaluation, all weights are frozen and each stream starts from an empty repository and profile (Appendix A).

4.2 Transferability across Executors (RQ1)

Executor-specific curators do not transfer well. Among the three SkillOS curators, the one trained with a given executor performs best with that executor on every benchmark (Table 1, upper). Mismatched pairings trail the matched curator by 2.3–10.0 pp. On ALFWorld, for example, SkillOS-γ reaches 41.1% with Qwen3-8B, compared with 50.0% for SkillOS-α, whereas SkillOS-α reaches 59.6% with GPT-OSS-120B, compared with 69.6% for SkillOS-γ (see also Figure 1). These gaps motivate conditioning curation decisions on the executor that consumes the skills.

Seen executors. A single EASE curator obtains the best average score on all three training executors, improving on the matched specialist by 2.7, 2.0, and 1.9 pp, respectively. It is best on ALFWorld (+5.0, +3.4, and +3.5 pp) and ScienceWorld (+0.9, +2.9, and +3.1 pp) for all three executors. On WebShop, it is best with Qwen3-8B, but trails the matched specialist by 0.4 and 0.9 pp with Qwen3-32B and GPT-OSS-120B, while still outperforming all training-free baselines. EASE thus matches or exceeds executor-specific curators with a single curator.

Unseen executors. Without further training, EASE also obtains the best average score on all three held-out executors, exceeding the strongest baseline, SkillOS-*best*, by 3.8, 2.7, and 2.1 pp on Kimi K2.6, DeepSeek V4 Flash, and Gemini 3.5 Flash, respectively (Table 1, lower). It is best in eight of the nine benchmark–executor combinations; the exception is ScienceWorld with Gemini 3.5 Flash, where it trails SkillOS-*best* by 0.5 pp. On ALFWorld, its margins over the strongest baseline are 4.8, 3.2, and 2.9 pp. Since SkillOS-*best* is selected separately for every executor and benchmark, these margins are measured against an optimistically selected reference. These results support transfer beyond the training pool.

4.3 Online Adaptation at Deployment (RQ2)

After a 64-task warm-up, continuing to update the profile improves ALFWorld SR over freezing it by 3.8, 2.2, and $1.1\,pp$ for Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively, with skill curation enabled in both conditions. Continued online curation with profile updates also outperforms a frozen repository by 5.9, 3.7, and $2.4\,pp$. A 16-task profile window outperforms both a 4-task window and full-history aggregation, while 16- and 64-task windows perform similarly (Appendix H, Table 6). Across six task orderings of the same task set, SR varies by at most $3.3\,pp$ (Appendix K). These results support keeping both behavioral evidence and skills current during deployment.

Table 1: Main results on ALFWorld (ALF), ScienceWorld (SciW), and WebShop (WS). We report category-macro success rate (%) on ALF and the benchmark score on SciW and WS; Avg. is their arithmetic mean. *Upper*: the three executors in the curator's training pool (α, β, γ). *Lower*: three unseen executors; SkillOS-*best* reports, per executor and benchmark, the highest score among SkillOS-α, SkillOS-β, and SkillOS-γ. **Bold** and underline mark the best and second-best results.

Method	QWEN3-8B (α)				QWEN3-32B (β)				GPT-OSS-120B (γ)			
	ALF	SciW	WS	Avg.	ALF	SciW	WS	Avg.	ALF	SciW	WS	Avg.
Training-Free												
No Memory	39.6	24.9	33.3	32.6	46.3	32.1	41.5	40.0	51.9	35.1	43.6	43.5
AWM	41.0	24.3	30.7	32.0	47.1	33.5	41.9	40.8	50.1	36.4	44.8	43.8
ReasoningBank	43.3	25.7	35.4	34.8	49.8	35.1	43.4	42.8	53.7	**38.1**	**46.8**	**46.2**
MemP	42.7	**26.4**	**35.7**	**34.9**	48.2	**36.8**	**45.3**	43.4	**54.1**	37.4	45.3	45.6
Memento-Skills	**44.9**	25.1	34.3	34.8	**50.1**	36.4	44.8	**43.8**	53.2	37.6	44.7	45.2
Finetuned Curators												
SkillOS-α	50.0	30.6	40.6	40.4	57.8	33.1	44.7	45.2	59.6	37.7	44.5	47.3
SkillOS-β	45.0	25.3	36.4	35.6	66.3	39.2	**49.3**	51.6	63.7	40.6	47.4	50.6
SkillOS-γ	41.1	22.1	32.1	31.8	61.2	36.9	45.1	47.7	69.6	43.3	**51.2**	54.7
EASE (*ours*)	**55.0**	**31.5**	**42.8**	**43.1**	**69.7**	**42.1**	48.9	**53.6**	**73.1**	**46.4**	50.3	**56.6**

Method	KIMI K2.6				DEEPSEEK V4 FLASH				GEMINI 3.5 FLASH			
	ALF	SciW	WS	Avg.	ALF	SciW	WS	Avg.	ALF	SciW	WS	Avg.
Training-Free												
No Memory	72.1	63.2	45.3	60.2	76.3	70.6	42.5	63.1	78.2	72.2	49.6	66.7
AWM	71.2	63.8	46.1	60.4	75.5	72.4	44.8	64.2	77.9	74.7	50.8	67.8
ReasoningBank	75.1	65.2	48.1	62.8	78.3	74.0	45.9	66.1	80.1	75.3	53.2	69.5
MemP	**76.3**	64.1	**48.5**	**63.0**	78.8	73.9	46.1	66.3	79.2	**77.1**	**55.6**	**70.6**
Memento-Skills	74.9	**65.7**	47.9	62.8	**79.4**	**74.5**	**46.3**	**66.7**	**80.7**	76.2	54.9	**70.6**
Finetuned Curators												
SkillOS-*best*	75.3	70.2	51.3	65.6	82.1	76.3	49.1	69.2	85.3	**79.1**	54.7	73.0
EASE (*ours*)	**81.1**	**72.6**	**54.6**	**69.4**	**85.3**	**78.4**	**51.9**	**71.9**	**88.2**	78.6	**58.5**	**75.1**

4.4 SKILL UTILITY AND EFFICIENCY (RQ3)

Performance and inference cost. Figure 3a shows that, on ALFWorld, EASE improves SR by $3.4 \sim 5.0$ *pp* over the matched SkillOS specialists while using $4.5 \sim 17.1\%$ fewer total deployment-time inference tokens (10.7% fewer in aggregate over the three executors). The savings come mainly from shorter skill context ($32.1 \sim 41.7\%$ fewer tokens) and fewer environment steps, which more than offset the $150 \sim 250$ extra curator tokens spent on the descriptor (Figure 3b; Appendix I).

Retrieval and functional benefit. Averaged equally over Qwen3-8B, Qwen3-32B, and GPT-OSS-120B on ALFWorld, EASE reduces mean repository size from 57.7 to 34.0 skills (41.0% fewer) and raises the retrieval rate of edited skill versions from 48.6% to 66.7% (37.3% relative) compared with the matched SkillOS specialists. Paired edit-rollback diagnostics show a mean local SR benefit of 2.67 pp for EASE edits versus 1.67 pp for the specialists (60.0% relative). The same trends, including lower token usage, hold on ScienceWorld and WebShop. Full measurements, definitions, and Figure 5 appear in Appendices I and J.

4.5 ABLATION STUDY (RQ4)

All variants in Figure 4 use the same executor pool and training budget.

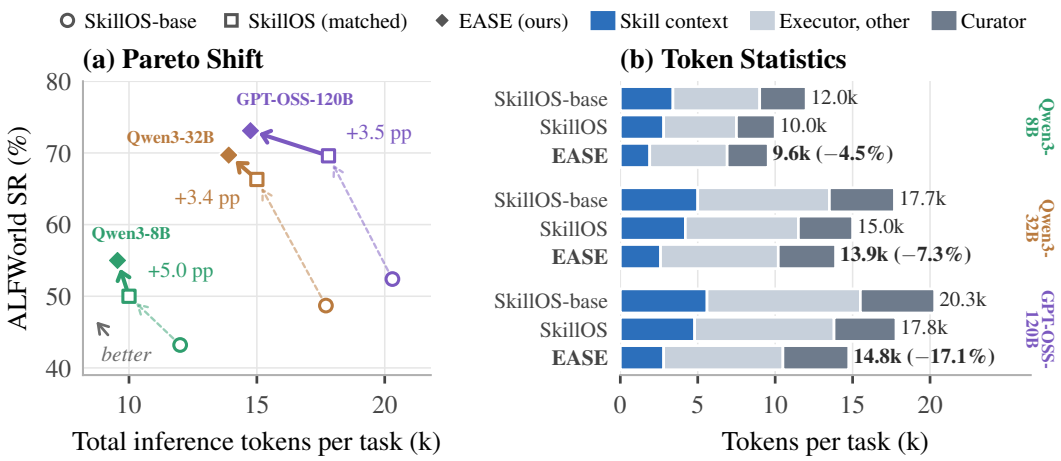


Figure 3: **Higher success with fewer inference tokens on ALFWorld.** (a) ***Pareto shift***: SR versus total inference tokens per task. (b) ***Token statistics***: per-task token composition; percentages give the change in total tokens relative to the matched specialist.

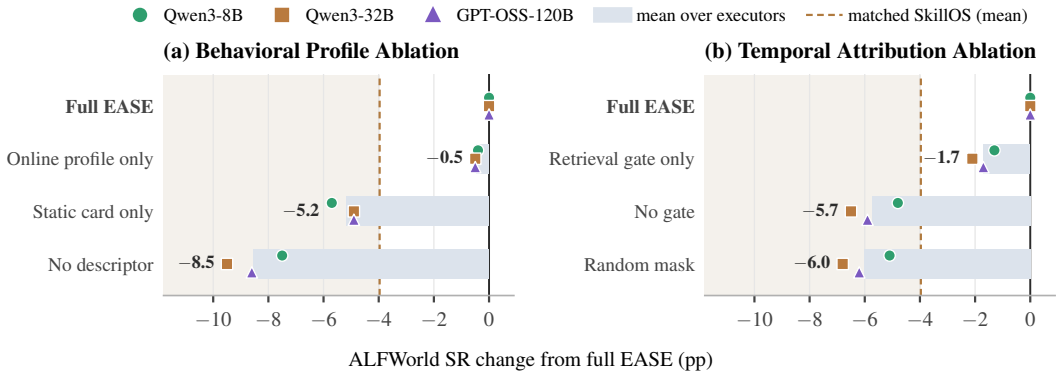


Figure 4: **Component ablations on ALFWorld.** Markers show the perf. drop of each ablation variant from full EASE (Table 1); The dashed line marks the mean change of the matched SkillOS specialists relative to EASE ($-4.0\,pp$).

Behavioral profile ablation. Removing the descriptor causes the largest drop, $7.5 \sim 9.5\,pp$, and places the pooled curator below every matched SkillOS specialist. A static model card recovers only $1.8 \sim 4.6\,pp$ of this loss and still trails the specialists, whereas the online profile alone comes within $0.5\,pp$ of the full descriptor (Figure 4a; Table 5). The benefit of conditioning thus comes mainly from observed behavior rather than from executor identity.

Temporal attribution ablation. Removing both gates costs $4.8 \sim 6.5\,pp$. The retrieval gate recovers most of this loss ($3.5 \sim 4.4pp$), and the behavioral gate adds a further $1.3 \sim 2.1\,pp$. A random mask with the same admission rate performs on par with ungated training, so the gain comes from *which* curation steps are updated (Figure 4b).

5 Conclusion

We presented EASE, a framework for behavior-adaptive skill curation. Unlike previous methods, EASE explicitly conditions on each executor's online behavior to curate customized skills at test time. Our experiments, across three benchmarks and six executors, have shown that EASE achieves the best average performance, improving on the strongest existing baseline by $1.9 \sim 3.8\,pp$, *without* per-executor finetuning. Beyond performance, EASE curates more compact skill repository whose edited skills are retrieved $36.3 \sim 38.7\%$ more often, while reducing total inference tokens by $9.1 \sim 14.5\%$ across all three benchmarks.

AI USE STATEMENT

We utilized AI tools for the following scope:

- **Writing & Polishing:** Fixing grammatical errors, improving readability, and rephrasing original text.
- **Feedback & Review:** Collecting feedback to improve paper structure and clarity.
- **LaTeX & Formatting:** Debugging LaTeX code for rendering with the required conference paper template.

All scientific claims, experimental execution, and analysis were conducted independently by human authors.

ETHICS STATEMENT

This work focuses on methodological advances in machine learning and uses publicly available benchmark datasets. We do not conduct experiments involving human subjects or collect personally identifiable information. We are not aware of any specific ethical concerns beyond those generally associated with the application of machine learning methods.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method and training procedure in Sections 3 and 4. Implementation details, hyperparameter settings and additional experimental configurations are provided in Appendix A. The source code required to reproduce the experiments is included in the anonymous supplementary material.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Anthropic. Equipping agents for the real world with agent skills, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Published October 16, 2025.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. MemP: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2026. arXiv:2508.06433.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. OS agents: A survey on MLLM-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7436–7465, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.369. URL https://aclanthology.org/2025.acl-long.369/.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. SkillGen: Verified inference-time agent skill synthesis, 2026. URL https://arxiv.org/abs/2605.10999.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. SkillOS: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026a.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*, 2026b. arXiv:2509.25140.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. SkillX: Automatically constructing skill knowledge bases for agents, 2026a. URL https://arxiv.org/abs/2604.04804.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. URL https://arxiv.org/abs/2305.16291.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1529–1550, San Diego, California, United States, July 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.69. URL https://aclanthology.org/2026.acl-long.69/.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022. doi: 10.48550/arXiv.2203.07540.

Yixuan Wang, Yiyang Zhou, Yiming Liang, Congyu Zhang, Fuxiao Liu, Jiawei Zhou, and Huaxiu Yao. Not all skills help: Measuring and repairing agent knowledge, 2026c. URL https://arxiv.org/abs/2606.15390.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue (Steve) Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification, 2026a. URL https://arxiv.org/abs/2604.01687.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://arxiv.org/abs/2510.04618.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization, 2026c. URL https://arxiv.org/abs/2606.08755.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-Skills: Let agents design agents, 2026. URL https://arxiv.org/abs/2603.18743.

A Extra Implementation Details

The implementation follows the online loop in Section 3: the executor completes a task before the profile and repository are updated, and the resulting skills are available only to later tasks.

Hyperparameters During RL finetuning, each GRPO update samples 32 groups, choosing one executor per group with probability $1/3$ and collecting eight rollouts over the same ordered stream of ten related training tasks. We retrieve at most $k = 5$ skills and use a profile window of $W = 16$ unless varied in the profile diagnostics. The paragraphs below describe the stream construction, reward settings, and optimization hyperparameters. At evaluation, the curator's weights are frozen and skills are edited online under the streaming protocol in §3.1. Kimi K2.6, DeepSeek V4 Flash, and Gemini 3.5 Flash are used only for transfer evaluation and are excluded from both training and checkpoint selection. Tables 2 and 3 collect the settings used for EASE.

Constructing experience for skill reuse. We use ALFWorld's six task categories to identify related training tasks. Each training stream contains ten distinct tasks from one category, presented in random order; tasks may be reused across streams. For each group, we sample Qwen3-8B, Qwen3-32B, or GPT-OSS-120B with equal probability and keep that executor and task sequence fixed across the eight rollouts. Each rollout starts with an empty repository and profile, so its editing decisions depend only on its own execution history. Since the training stream length $T = 10$ is shorter than the profile window $W = 16$, training profiles aggregate all completed tasks in the current rollout. During longer evaluation streams, the window retains only the most recent 16 completed tasks.

Table 2: Method and evaluation settings. The profile window is varied only in the profile-dynamics diagnostics. ALFWorld evaluation tasks are drawn from its 134-task `valid unseen` split.

Setting	Value
Retrieved skills, k	5
Profile window, W	16
Rollouts per group, N	8
Training tasks per stream, T	10
Skill-free references per task and executor, m	4
Behavioral-deviation threshold, τ	0.05
Function-call reward weight, λ_f	1.0
Compression reward weight, λ_c	0.05
Quality reward weight, λ_u	0.10
Environment actions per episode (maximum)	50
Task-order permutations	6
Profile-diagnostic warm-up, t_0	64
Profile-diagnostic suffix length	70
Retrieval follow-up window, H	20
Edit-rollback paired comparisons per executor	200

Learning the transferable curator. We optimize LoRA parameters of Qwen3-8B with AdamW, keeping the underlying model and all executors frozen. The adapters have rank 16, scaling factor 32, and dropout 0.05, and are applied to the attention and feed-forward projections listed in Table 3. Each sampling batch contains $B = 32$ executor–stream groups, each with $N = 8$ rollouts; B therefore counts groups before rollout expansion. We perform one optimization epoch per rollout batch for 60 updates. The objective is the masked surrogate in Eq. (11), without a KL penalty. Advantages are normalized within each group using $\epsilon_A = 10^{-6}$, before the gates select editing steps for optimization.

Supervising edits through outcomes and quality. The task reward averages ALFWorld's binary environment-success signals on $x_2, \ldots, x_T$. A frozen Qwen3-32B judge evaluates each editing step once at temperature zero, assessing correctness, executability, and reusability on a $[0, 1]$ scale. These quality scores are averaged over editing steps and combined with the other reward terms

Table 3: Curator optimization and quality-judge settings. Executor models and the judge remain frozen throughout training; only the curator's LoRA parameters are optimized.

Setting	**Value**
Curator backbone	Qwen3-8B
Finetuning method	LoRA
LoRA rank	16
LoRA scaling factor	32
LoRA dropout	0.05
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
Optimizer	AdamW
Learning rate	1×10^{-6}
Groups per sampling batch, B	32
Clipping parameter, ε	0.2
Advantage normalization constant, ϵ_A	1×10^{-6}
KL coefficient, β	0
Curator training temperature	1.0
Maximum input length (tokens)	16,384
Maximum output length (tokens)	4,096
Training updates	60
Optimization epochs per rollout batch	1
Quality-judge backbone	Qwen3-32B
Judge temperature	0
Judge evaluations per editing step	1
Judge score range	$[0, 1]$

using the weights in Table 2. For the behavioral gate, we cache four skill-free reference trajectories per training task and executor, using the same initial state and execution configuration as the skill-assisted runs, and set $\tau = 0.05$. Reference trajectories and judge scores provide training supervision only.

Following adaptation at evaluation. We use BM25 as retriever throughout this paper over skill keywords (Robertson & Zaragoza, 2009), although EASE is not tied to a particular retriever. Each executor is evaluated on a stream of all 134 `valid unseen` tasks, with an empty repository and profile at the start. The task-order analysis uses six permutations of a 120-task subset of this split (Appendix K). ALFWorld success rate is macro-averaged over task categories:

$$\mathrm{SR}_{\text{macro}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{D}_c|} \sum_{i \in \mathcal{D}_c} \mathbf{1}[\text{task } i \text{ succeeds}], \tag{7}$$

where $\mathcal{C}$ is the set of categories and $\mathcal{D}_c$ is the set of tasks in category c. For the profile diagnostics, conditions branch from a common history after task 64 and are scored on the remaining tasks, isolating continued adaptation from the shared warm-up. Skill retrieval is measured over the next $H = 20$ tasks for versions with a complete follow-up window. The edit-rollback diagnostics use paired comparisons per executor on tasks from the same evaluation split; these additional executions do not update the main evaluation history (Appendix J). The ScienceWorld and WebShop efficiency and skill-utility measurements follow the same protocol on a 100-task evaluation stream per executor. Edit-rollback diagnostics sample 50 edits per method and executor, each paired on four tasks, and differences are measured in benchmark score points.

B Executor behavioral profile features

Table 4 lists the feature families of $P_t(E)$ (§3.2). Features use stored logs and environment signals without additional model calls. Their availability depends on the environment: an action rejection

may be directly observed, whereas a semantic error type or implicit reliance on a skill may not be. Unavailable features are marked as missing. Self-assessed correctness is recorded separately from externally observed outcomes.

For the window in Eq. (2), rates pool event counts and their corresponding exposure counts, rather than averaging episode-level percentages with unequal denominators. A feature with no eligible observations is unavailable, not zero. Averages carry their sample counts, and the profile records the number of completed episodes in its window. Updating the profile after x_t ensures that the edit Δ_t uses observations through t only.

Table 4: Executor behavioral profile features. Examples are included only when supported by observable signals; retrieval alone is not evidence that a skill was followed.

Family	Example statistics and observability
Action validity	Rejected-action rate and parse failures from action validation.
Failure recovery	Repeated rejected actions; steps to the next accepted action, with unrecovered episodes counted separately.
State tracking	Repeated visits to logged states; precondition violations when exposed by the environment.
Efficiency	Steps per episode and frequency of reaching the interaction budget.
Error types	Subtask failures or reward-component losses when the environment exposes them.
Skill utilization	Retrieval frequency; explicit skill references or invocations when logged; outcomes conditional on retrieval or observed use, reported separately.

C Executor composition of GRPO groups

Within each GRPO group, all N rollouts share the same frozen executor and ordered task stream. Their repositories, profiles, and sampled execution histories evolve independently. The mean in Eq. (4) therefore compares curation policies under a common executor and task sequence, avoiding direct comparisons between the base success rates of different executors within the same advantage calculation. It does not remove executor sampling noise, environment stochasticity, or differences between repository histories.

The executor is sampled across groups to provide experience from multiple executors. Since even one executor's profile changes over time, diversity is an additional source of variation in conditioning, not a prerequisite for the descriptor to vary. Nor does diversity force the curator to use the profile; removing or shuffling it provides an empirical test. Executors near a benchmark's floor or ceiling may offer little variation in task outcomes, but the resulting gradient magnitude also depends on auxiliary rewards and the normalization by $\sigma_G + \epsilon_A$. Identical total rewards within a group yield zero advantages.

D Credit Assignment

Gate 1: retrieval incidence. For $j > t$, let $\mathcal{T}_{t,j}$ contain the before- and after-edit versions of skills changed by Δ_t that have not been edited again before x_j. Let $\Pi_{t,j}$ restore their pre-edit state in the current repository $\mathcal{S}_j$, leaving other skills unchanged. We define

$$\begin{aligned} \rho_{t,j}^{(n)} &= \mathbf{1}\Big[\mathcal{T}_{t,j}^{(n)} \cap \Big(\mathcal{R}_k(x_j; \mathcal{S}_j^{(n)}) \cup \mathcal{R}_k(x_j; \Pi_{t,j}(\mathcal{S}_j^{(n)}))\Big) \neq \varnothing\Big], \\ Z_t^{(n)} &= \sum_{j=t+1}^{T} \rho_{t,j}^{(n)}. \end{aligned} \tag{8}$$

The patched retrieval permits credit for deleting a skill that would otherwise have been retrieved. Tracking versions avoids silently attributing a later rewrite to an earlier edit; it also omits effects inherited through subsequent edits. Appendix E specifies the patch and operation-wise tests. Gate 1 admits steps with $Z_t^{(n)} > 0$.

Gate 2: behavioral deviation. Retrieval alone does not establish that a skill changes execution. For each training task and executor, we cache $m \geq 2$ skill-free action sequences $\{\tilde{a}_j^{(r)}\}_{r=1}^{m}$, using the same initial environment state, prompting, and decoding configuration as skill-assisted execution. We measure deviation from these references as

$$\mathcal{D}_j^{(n)} = \left[\frac{1}{m} \sum_{r=1}^{m} d(a_j^{(n)}, \tilde{a}_j^{(r)}) - \binom{m}{2}^{-1} \sum_{r<r'} d(\tilde{a}_j^{(r)}, \tilde{a}_j^{(r')}) \right]_+, \tag{9}$$

where $[z]_+ = \max(z, 0)$ and d is the Levenshtein distance between action sequences divided by $\max(|a|, |b|, 1)$. Each parsed action is one sequence element. The second term estimates variation among skill-free runs. Moreover, $\mathcal{D}_j$ measures deviation under the entire retrieved set, not improvement or the marginal effect of Δ_t.

Combined admission. With both gates enabled, the admitted set is

$$\mathcal{M}^{(n)} = \left\{ t : Z_t^{(n)} > 0 \text{ and } \max_{j > t : \rho_{t,j}^{(n)} = 1} \mathcal{D}_j^{(n)} > \tau \right\}, \qquad \tau \geq 0. \tag{10}$$

The maximum is evaluated only when $Z_t^{(n)} > 0$. Disabling Gate 2 explicitly removes the second condition; setting $\tau = 0$ still excludes zero deviation. Disabling both gates admits all curation steps. Reference trajectories are reused while their executor and environment configuration remains unchanged, with their inference cost accounted for as training overhead.

Masked policy optimization. Let $y_{t,\ell}^{(n)}$ be token ℓ of the curator's generated response at step t, of length $L_t^{(n)}$, and let $h_{t,\ell}^{(n)}$ be its logged context, including preceding output and function-call feedback. The token importance ratio is $q_{t,\ell}^{(n)}(\theta) = \pi_\theta(y_{t,\ell}^{(n)} \mid h_{t,\ell}^{(n)}) / \pi_{\theta_{\text{old}}}(y_{t,\ell}^{(n)} \mid h_{t,\ell}^{(n)})$. Using the clipped surrogate of proximal policy optimization (Schulman et al., 2017), we maximize the following masked objective over $\mathcal{I} = \{n : |\mathcal{M}^{(n)}| > 0\}$:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{I}|} \sum_{n \in \mathcal{I}} \frac{1}{|\mathcal{M}^{(n)}|} \sum_{t \in \mathcal{M}^{(n)}} \frac{1}{L_t^{(n)}} \sum_{\ell=1}^{L_t^{(n)}} \ell_{\text{clip}}\left(q_{t,\ell}^{(n)}(\theta), A^{(n)} \right), \tag{11}$$

$$\ell_{\text{clip}}(q, A) = \min(qA, \text{clip}(q, 1-\varepsilon, 1+\varepsilon)A).$$

Empty admitted sets are omitted from the loss, and the update is skipped if $\mathcal{I}$ is empty; advantages are always computed from the original N rollouts. Masks are held fixed during optimization. This applies the same composite advantage to every admitted step, so masking also removes direct auxiliary-reward supervision for excluded steps, including the final edit when no subsequent task is available. We retain this tradeoff explicitly rather than treating the gates as exact credit assignment.

E RETRIEVAL-INCIDENCE TESTS

The patch in Eq. (8) is a local repository intervention, not a replay of the stream without an earlier edit. We record skill identifiers and before- and after-edit versions for each successfully applied operation. For a sequence of calls Δ_t, the relevant change is its net effect on each identifier; failed calls and sequences with no net repository change contribute no touched versions.

For a later task x_j, an identifier remains eligible for the test only if no successful operation has touched it between steps $t+1$ and $j-1$. Let $U_{t,j}$ be the eligible identifiers whose state changed at t. The set $\mathcal{T}_{t,j}$ contains their nonempty pre- and post-edit versions. The patch $\Pi_{t,j}$ replaces the current state of every identifier in $U_{t,j}$ with its pre-edit state, leaving all other entries unchanged. Restoring an absent pre-edit state removes an added skill; restoring a deleted skill inserts its saved version. Version comparisons include the identifier, header, and body. This rule excludes superseded edits from direct attribution, including cases where a later edit reinstates identical text.

For an isolated change that remains eligible at j, the tests reduce to

$$\begin{aligned} \texttt{add}(s): &\quad s \in \mathcal{R}_k(x_j; \mathcal{S}_j), \\ \texttt{delete}(s): &\quad s \in \mathcal{R}_k(x_j; \mathcal{S}_j \cup \{s\}), \\ \texttt{modify}(s \to s'): &\quad s' \in \mathcal{R}_k(x_j; \mathcal{S}_j) \text{ or } s \in \mathcal{R}_k(x_j; (\mathcal{S}_j \setminus \{s'\}) \cup \{s\}). \end{aligned}$$

A step containing several changes uses the joint patch defined above. Both actual and patched retrieval use the repository at j, rather than the stale repository at t, and use the same retrieval settings and tie-breaking rule. Actual retrieval is already logged; evaluating a nontrivial joint patch requires one additional retrieval query per (t, j) pair. Across a stream, the number of such queries can grow quadratically with its length. These checks add no executor calls, but their retrieval and version-storage costs should still be measured.

F Compression reward under executor conditioning

Let χ_t be the curator input and let χ_t^{base} be the same input with the descriptor field e_t removed. Writing len_{tok} for token length under a fixed tokenizer, the rollout-level compression term is

$$r_{\text{comp}}^{(n)} = \frac{1}{T} \sum_{t=1}^{T} \left(1 - \frac{\text{len}_{\text{tok}}(\mathcal{S}_{t+1}^{(n)})}{\max\{1, \text{len}_{\text{tok}}(\chi_t^{\text{base},(n)})\}} \right). \tag{12}$$

The numerator measures the repository after applying Δ_t; the denominator measures the descriptor-free input used to produce it. Excluding both the profile and static metadata prevents their length from directly increasing this reward. The term remains a relative compression measure: differences in trajectories and retrieved context can still change its denominator. It is neither a fixed token budget nor a direct measurement of retrieval cost, which must be reported separately.

G Curator Conditioning Comparisons

Conditioning a skill curator. Table 5 summarizes the conditioned curator configurations. These results complement the executor-specific references in Table 1.

Table 5: Conditioned curator comparisons on ALFWorld. Entries are category-macro SR (%). All configurations enable both temporal-attribution gates; the first two rows correspond to the "No descriptor" and "Static card only" variants in Figure 4a.

Configuration	**Executors**		
	QWEN3-8B	QWEN3-32B	GPT-OSS-120B
Pooled, unconditioned	47.5	60.2	64.5
Static-card conditioning	49.3	64.8	68.2
EASE (*ours*)	**55.0**	**69.7**	**73.1**

Pooling helps, but leaves a gap to executor-specific training. The unconditioned pooled curator improves over SkillOS-base by 4.3, 11.5, and 12.1 pp on Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively, but remains 2.5, 6.1, and 5.1 pp below the corresponding specialists. Exposure to trajectories from multiple executors alone therefore does not close the gap. Adding a static model card raises SR by a further 1.8, 4.6, and 3.7 pp, yet the static-card variant still trails the matched specialists by 0.7, 1.5, and 1.4 pp. Identifying the executor helps, but identity alone is not sufficient.

The complete EASE configuration improves on the unconditioned pooled curator by 7.5, 9.5, and 8.6 pp, and on the static-card variant by 5.7, 4.9, and 4.9 pp. Since all three configurations enable both temporal-attribution gates, the difference between the static-card variant and the complete configuration isolates the contribution of the online profile, which is what lifts the shared curator above the executor-specific specialists.

H Online Adaptation and Profile Dynamics

Recent behavior and continued profile updates. Table 6 examines deployment-time adaptation using the EASE curator evaluated in Table 1. After a common warm-up ending at task $t_0 = 64$, each condition starts from a copy of the same repository and interaction history and follows the same remaining task order. Freezing the profile while continuing skill edits tests the value of fresh behavioral evidence; freezing the repository tests the value of continued online editing. The remaining conditions compare profile windows of 4, 16, and 64 tasks with full-history aggregation. Curator and executor weights remain fixed throughout. Scores are category-macro SR on the common 70-task suffix of the 134-task `valid unseen` stream. These scores are reported separately from the full-stream main evaluation; with the reference configuration, SR on the 70-task suffix exceeds SR on the 64-task warm-up by 6.3, 5.4, and 4.4 pp on Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, consistent with the benefit of accumulated experience. Additional task-order diagnostics are reported in Appendix K.

Updating the profile with $W = 16$ exceeds freezing it at t_0 by 3.8, 2.2, and 1.1 pp, respectively. Since both conditions continue editing skills, the comparison supports the value of keeping behavioral evidence current as the repository evolves. Relative to a frozen repository, the reference configuration gains 5.9, 3.7, and 2.4 pp. These are complementary deployment interventions, rather than additive estimates of profile and repository effects.

The reference window also exceeds full-history aggregation by 1.3, 0.9, and 0.5 pp and the four-task window by 1.7, 1.0, and 0.6 pp. This pattern is consistent with retaining enough recent experience to characterize recurring behavior while limiting the influence of older interactions. The 16- and 64-task windows are close: $W = 64$ leads by 0.3 pp on Qwen3-8B and 0.1 pp on GPT-OSS-120B, while $W = 16$ leads by 0.2 pp on Qwen3-32B. The results support continued profile updates and a finite recent history, without identifying one window as uniformly best across executors.

Table 6: Profile-dynamics diagnostics. Entries report category-macro SR (%) on the common 70-task suffix following a 64-task warm-up in the 134-task `valid unseen` stream. Except for the frozen-repository control, all conditions continue editing skills. The short, reference, and long windows contain 4, 16, and 64 completed tasks, respectively.

Configuration	Executors		
	QWEN3-8B	QWEN3-32B	GPT-OSS-120B
Updating profile, $W_{\text{ref}} = 16$	58.0	**72.3**	75.2
Profile frozen at t_0, $W_{\text{ref}} = 16$	54.2	70.1	74.1
Updating profile, $W_{\text{short}} = 4$	56.3	71.3	74.6
Updating profile, $W_{\text{long}} = 64$	**58.3**	72.1	**75.3**
Updating profile, full history	56.7	71.4	74.7
Repository frozen at t_0	52.1	68.6	72.8

I Additional Efficiency Measurements

Table 7 provides the absolute measurements underlying Figure 3, together with skill-context and profile/card token counts and environment steps. Token counts and steps are averages over all evaluation tasks, including failures, with runtime settings held fixed within each executor comparison. Curator and executor counts each include input and output tokens. Skill-context tokens are part of executor input, and profile/card tokens are part of curator input; neither is added again to the total. Environment steps measure execution effort. Reference-bank generation and gate computation are training overhead and are excluded from these deployment measurements.

Relative to the matched specialists, EASE uses 150, 200, and 250 more curator tokens per task, but 600, 1,300, and 3,300 fewer executor tokens on Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively, reducing total tokens by 4.5%, 7.3%, and 17.1% (10.7% in aggregate). Skill-context tokens decrease by 32.1%, 38.1%, and 41.7%, and environment steps decrease for all three executors. These counts do not measure latency, monetary cost, or the additional cost of training.

Table 8 reports the same measurements on ScienceWorld. EASE uses 250 more curator tokens per task than the matched specialists, but 4,000, 5,000, and 6,000 fewer executor tokens on Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively, reducing total tokens by 13.4%, 14.7%, and 15.1% (14.5% in aggregate). Skill-context tokens decrease by 32.3%, 33.3%, and 34.1%, and environment steps again decrease for all three executors. Table 9 reports the corresponding WebShop measurements. EASE again uses 250 more curator tokens per task, but 3,500, 3,500, and 3,000 fewer executor tokens, reducing total tokens by 10.8%, 9.6%, and 7.4% (9.1% in aggregate); skill-context tokens decrease by 30.8%, 30.6%, and 31.3%, and environment steps decrease for all three executors. On WebShop, these savings are not accompanied by a higher score on every executor: EASE trails the matched specialist by 0.4 and 0.9 points with Qwen3-32B and GPT-OSS-120B (Table 1).

Table 7: Performance and inference-token measurements on ALFWorld. SR for SkillOS and EASE matches Table 1. Token counts and steps are per-task averages; skill-context and descriptor tokens are included in the curator/executor totals. Dashes denote the absence of a profile/card input.

Metric	**SkillOS-base**	**SkillOS**	**EASE**
Executor: QWEN3-8B			
SR (%)	43.2	50.0	55.0
Curator tokens / task	3k	2.5k	2.65k
Executor tokens / task	9k	7.5k	6.9k
Skill-context tokens / task	3.4k	2.8k	1.9k
Profile/card tokens / task	-	-	245
Environment steps / task	28.5	25.0	22.0
Executor: QWEN3-32B			
SR (%)	48.7	66.3	69.7
Curator tokens / task	4.2k	3.5k	3.7k
Executor tokens / task	13.5k	11.5k	10.2k
Skill-context tokens / task	5k	4.2k	2.6k
Profile/card tokens / task	-	-	255
Environment steps / task	27.5	24.0	21.5
Executor: GPT-OSS-120B			
SR (%)	52.4	69.6	73.1
Curator tokens / task	4.8k	4k	4.25k
Executor tokens / task	15.5k	13.8k	10.5k
Skill-context tokens / task	5.6k	4.8k	2.8k
Profile/card tokens / task	-	-	260
Environment steps / task	26.5	23.0	20.5

J SKILL UTILITY MEASUREMENTS

Table 10 provides detailed executor-level measurements underlying Figure 5. The figure takes the arithmetic mean of the three executor entries for each method and metric. For fair comparison, the method ordering is the same within each executor for all three metrics. Tables 11 and 12 report the same measurements on ScienceWorld and WebShop, where the edit-rollback benefit is measured in score points. Averaged over executors, EASE reduces repository size relative to the matched specialists by 41.0%, 37.7%, and 34.5% on ALFWorld, ScienceWorld, and WebShop, raises the version retrieval rate by 37.3%, 38.7%, and 36.3% (relative), and increases the edit-rollback benefit by 60.0%, 51.8%, and 58.5% (relative).

For each added or modified skill version with a complete $H = 20$-task follow-up window, we record whether that version is retrieved at least once during the window. Specifically, retrieval of a later rewrite is not credited to an earlier version. Versions with fewer than 20 subsequent tasks available in the evaluation stream are excluded from this metric for fair comparison. Skill repository size is averaged over task boundaries before averaging across executors.

Table 8: Performance and inference-token measurements on ScienceWorld. Scores for SkillOS and EASE match Table 1. Token counts and steps are per-task averages; skill-context and descriptor tokens are included in the curator/executor totals. Dashes denote the absence of a profile/card input.

Metric	SkillOS-base	SkillOS	EASE
Executor: QWEN3-8B			
Score	25.8	30.6	31.5
Curator tokens / task	5.8k	5k	5.25k
Executor tokens / task	27k	23k	19k
Skill-context tokens / task	7.8k	6.2k	4.2k
Profile/card tokens / task	-	-	250
Environment steps / task	32.5	28.7	24.4
Executor: QWEN3-32B			
Score	34.0	39.2	42.1
Curator tokens / task	6.2k	5.4k	5.65k
Executor tokens / task	31k	27k	22k
Skill-context tokens / task	9.2k	7.5k	5k
Profile/card tokens / task	-	-	255
Environment steps / task	34.2	30.5	26.5
Executor: GPT-OSS-120B			
Score	37.0	43.3	46.4
Curator tokens / task	7k	6k	6.25k
Executor tokens / task	37k	32k	26k
Skill-context tokens / task	10.5k	8.5k	5.6k
Profile/card tokens / task	-	-	260
Environment steps / task	36.0	31.8	27.3

Table 9: Performance and inference-token measurements on WebShop. Scores for SkillOS and EASE match Table 1. Token counts and steps are per-task averages; skill-context and descriptor tokens are included in the curator/executor totals. Dashes denote the absence of a profile/card input.

Metric	SkillOS-base	SkillOS	EASE
Executor: QWEN3-8B			
Score	35.2	40.6	42.8
Curator tokens / task	6k	5.2k	5.45k
Executor tokens / task	28k	25k	21.5k
Skill-context tokens / task	8k	6.5k	4.5k
Profile/card tokens / task	-	-	250
Environment steps / task	18.5	16.0	13.5
Executor: QWEN3-32B			
Score	44.5	49.3	48.9
Curator tokens / task	6.5k	5.7k	5.95k
Executor tokens / task	32k	28k	24.5k
Skill-context tokens / task	9k	7.2k	5k
Profile/card tokens / task	-	-	255
Environment steps / task	19.5	17.0	14.5
Executor: GPT-OSS-120B			
Score	46.5	51.2	50.3
Curator tokens / task	7k	6.2k	6.45k
Executor tokens / task	36k	31k	28k
Skill-context tokens / task	10k	8k	5.5k
Profile/card tokens / task	-	-	260
Environment steps / task	21.0	18.5	16.0

To assess functional benefit, we conduct paired comparisons per executor on hold-out tasks for evaluation. Each comparison pairs an edited repository with the same repository after rolling back the selected edit. (Rollback removes an addition, restores the previous version for a modification, or reinstates a deleted skill.) The mean paired difference in task success, expressed in percentage points, measures the local benefit of the edit; diagnostic trajectories are not fed back into the main evaluation stream.

These local differences support the functional value of the evaluated edits; they do not decompose the full-stream performance gain or establish that every retrieved skill is beneficial. Together with the direct context measurements in Appendix I, Table 7, the results support compact repositories whose edits are more often retrieved and yield larger measured local benefits.

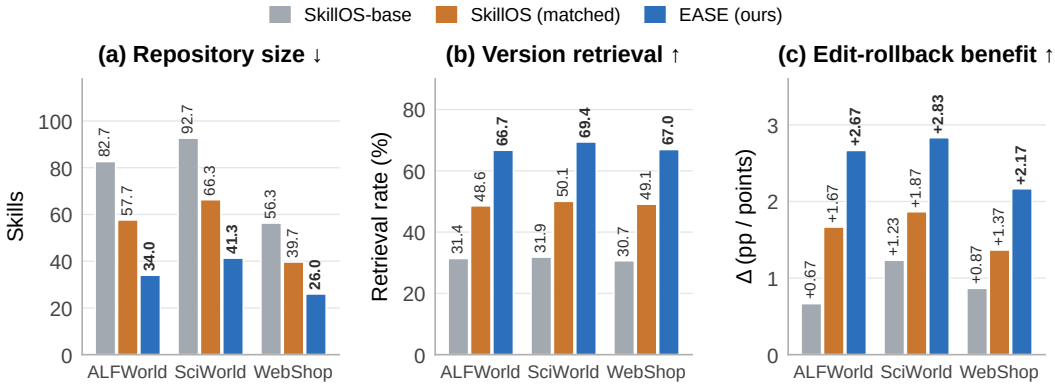


Figure 5: **Compact repositories with higher retrieval and edit utility.** Bars show equal-weight means across Qwen3-8B, Qwen3-32B, and GPT-OSS-120B on ALFWorld, ScienceWorld, and WebShop. SkillOS denotes the executor-matched specialists. Each panel retains its original units; the edit-rollback benefit is measured in SR percentage points on ALFWorld and in score points on ScienceWorld and WebShop; arrows indicate the preferred direction. Complete per-executor measurements and metric definitions are provided in Tables 10, 11, and 12.

Table 10: Skill utilization rate comparison on ALFWorld.

Metric	SkillOS-base	SkillOS	EASE
Executor: QWEN3-8B			
Version retrieval rate (%)	29.5	47.2	64.6
Edit-rollback ΔSR (*pp*)	+0.5	+1.5	+3.0
Repository size (skills)	85	60	35
Executor: QWEN3-32B			
Version retrieval rate (%)	31.4	48.6	67.4
Edit-rollback ΔSR (*pp*)	+0.5	+1.5	+2.5
Repository size (skills)	83	58	34
Executor: GPT-OSS-120B			
Version retrieval rate (%)	33.3	50.0	68.2
Edit-rollback ΔSR (*pp*)	+1.0	+2.0	+2.5
Repository size (skills)	80	55	33

K TASK-ORDER SENSITIVITY

Sensitivity to task order. To test whether EASE is sensitive to the order in which tasks arrive, we sample a fixed set S of 120 tasks covering all six categories of the ALFWorld held-out split and

Table 11: Skill utilization rate comparison on ScienceWorld. Retrieval rates are computed over 115/88/60, 108/82/56, and 104/78/54 complete-window versions (SkillOS-base/SkillOS/EASE) for Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively; edit-rollback benefits use 50 sampled edits per method and executor, each paired on four tasks.

Metric	**SkillOS-base**	**SkillOS**	**EASE**
Executor: QWEN3-8B			
Version retrieval rate (%)	30.4	48.9	68.3
Edit-rollback ΔScore (points)	+0.6	+1.1	+1.8
Repository size (skills)	98	70	44
Executor: QWEN3-32B			
Version retrieval rate (%)	31.5	50.0	69.6
Edit-rollback ΔScore (points)	+1.4	+2.0	+3.1
Repository size (skills)	92	66	41
Executor: GPT-OSS-120B			
Version retrieval rate (%)	33.7	51.3	70.4
Edit-rollback ΔScore (points)	+1.7	+2.5	+3.6
Repository size (skills)	88	63	39

Table 12: Skill utilization rate comparison on WebShop. Retrieval rates are computed over 95/76/55, 91/75/52, and 88/71/47 complete-window versions (SkillOS-base/SkillOS/EASE) for Qwen3-8B, Qwen3-32B, and GPT-OSS-120B, respectively; edit-rollback benefits use 50 sampled edits per method and executor, each paired on four tasks.

Metric	**SkillOS-base**	**SkillOS**	**EASE**
Executor: QWEN3-8B			
Version retrieval rate (%)	30.5	47.4	65.5
Edit-rollback ΔScore (points)	+0.6	+1.0	+1.7
Repository size (skills)	58	41	27
Executor: QWEN3-32B			
Version retrieval rate (%)	30.8	49.3	67.3
Edit-rollback ΔScore (points)	+0.9	+1.4	+2.2
Repository size (skills)	57	40	26
Executor: GPT-OSS-120B			
Version retrieval rate (%)	30.7	50.7	68.1
Edit-rollback ΔScore (points)	+1.1	+1.7	+2.6
Repository size (skills)	54	38	25

construct six evaluation streams that contain the same tasks in different orders, $P_t = \texttt{permute}_t(S)$ for $t \in \{1, \ldots, 6\}$. With the trained EASE curator and Qwen3-8B as the executor, each stream is solved independently from an empty repository and profile, without prior knowledge of the ordering. Table 13 reports the SR of each stream.

Table 13: Task-order analysis over six permutations of the same 120 ALFWorld tasks with Qwen3-8B as the executor. SR is the stream-level success rate (%).

Task order	1	2	3	4	5	6
SR	57.5	56.7	55.8	57.5	57.5	54.2

SR ranges from 54.2% to 57.5% (a spread of 3.3 pp) with no extreme outliers, suggesting that EASE is not overly sensitive to the specific ordering of tasks. Real-world task streams, however, can mix domains and difficulty levels in ways that are hard to model, and a more thorough study of ordering effects is left to future work.